INFO 6210

Data Management and Database Design

In Class Exam Solutions

Professor: Nik Bear Brown

Exam is Tuesday April 9, 2019

One page cheat sheet. No books, computer or phone.

All are 5 points unless noted otherwise.

1 (10 points)) Write a query in SQL to display the employee ID, name, salary, department name, location, department ID, job name of all the employees working at SYDNEY or working in the FINANCE department with an annual salary above 28000, but the monthly salary should not be 3000 or 2800 and who does not works as a MANAGER and whose ID containing a digit of '3' or '7' in 3rd position. List the result in ascending order of department ID and descending order of job name.

Sample table: employees

emp\_id | emp\_name | job\_name | manager\_id | hire\_date | salary | commission | dep\_id

Sample table: department

dep\_id | dep\_name | dep\_location

**Solution:**

SELECT E.emp\_id,

E.emp\_name,

E.salary,

D.dep\_name,

D.dep\_location,

E.dep\_id,

E.job\_name

FROM employees E,

department D

WHERE (D.dep\_location = 'SYDNEY'

OR D.dep\_name = 'FINANCE')

AND E.dep\_id=D.dep\_id

AND E.emp\_id IN

(SELECT emp\_id

FROM employees E

WHERE (12\*E.salary) > 28000

AND E.salary NOT IN (3000,

2800)

AND E.job\_name !='MANAGER'

AND (trim(to\_char(emp\_id,'99999')) LIKE '\_\_3%'

OR trim(to\_char(emp\_id,'99999')) LIKE '\_\_7%'))

ORDER BY E.dep\_id ASC,

E.job\_name DESC;

2 (10 points) Write a query in SQL to list all the employees of grade 2 and 3.

Sample table: employees

emp\_id | emp\_name | job\_name | manager\_id | hire\_date | salary | commission | dep\_id

Sample table: salary\_grade

grade | min\_sal | max\_sal

**Solution:**

SELECT \*

FROM employees e,

salary\_grade s

WHERE e.salary BETWEEN s.min\_sal AND s.max\_sal

AND s.grade IN (2, 3);

3 (10 points) Write a query in SQL to find the name of the venue with city where the EURO cup 2016 final match was played

Sample table: soccer\_venue

venue\_id | venue\_name | city\_id | aud\_capacity

----------+-------------------------+---------+--------------

Sample table: soccer\_city

city\_id | city | country\_id

---------+---------------+------------

Sample table: match\_mast

match\_no | play\_stage | play\_date | results | decided\_by | goal\_score | venue\_id | referee\_id | audence | plr\_of\_match | stop1\_sec | stop2\_sec

----------+------------+------------+---------+------------+------------+----------+------------+---------+--------------+-----------+-----------

**Solution:**

SELECT venue\_name, city

FROM soccer\_venue a

JOIN soccer\_city b ON a.city\_id=b.city\_id

JOIN match\_mast d ON d.venue\_id=a.venue\_id

AND d.play\_stage='F';

4) Write a query in SQL to find the number of goal scored by each team in every match within normal play schedule

Sample table: match\_details

match\_no | play\_stage | team\_id | win\_lose | decided\_by | goal\_score | penalty\_score | ass\_ref | player\_gk

----------+------------+---------+----------+------------+------------+---------------+---------+-----------

Sample table: soccer\_country

country\_id | country\_abbr | country\_name

------------+--------------+---------------------

**Solution:**

SELECT match\_no,country\_name,goal\_score

FROM match\_details a

JOIN soccer\_country b

ON a.team\_id=b.country\_id

WHERE decided\_by='Normal play schedule'

ORDER BY match\_no;

5) Write a query in SQL to find the highest individual scorer in EURO cup 2016.

Sample table: goal\_details

goal\_id | match\_no | player\_id | team\_id | goal\_time | goal\_type | play\_stage | goal\_schedule | goal\_half

---------+----------+-----------+---------+-----------+-----------+------------+---------------+-----------

Sample table: player\_mast

player\_id | team\_id | jersey\_no | player\_name | posi\_to\_play | dt\_of\_bir | age | playing\_club

-----------+---------+-----------+-------------------------+--------------+------------+-----+---------------------

Sample table: soccer\_country

country\_id | country\_abbr | country\_name

------------+--------------+---------------------

**Solution:**

SELECT player\_name,country\_name,count(player\_name)

FROM goal\_details gd

JOIN player\_mast pm ON gd.player\_id =pm.player\_id

JOIN soccer\_country sc ON pm.team\_id = sc.country\_id

GROUP BY country\_name,player\_name HAVING COUNT(player\_name) >= ALL

(SELECT COUNT(player\_name)

FROM goal\_details gd

JOIN player\_mast pm ON gd.player\_id =pm.player\_id

JOIN soccer\_country sc ON pm.team\_id = sc.country\_id

GROUP BY country\_name,player\_name);

6) Write a query in SQL to find the name and country of the referee who assisted the referee in the final match.

Sample table: asst\_referee\_mast

ass\_ref\_id | ass\_ref\_name | country\_id

------------+--------------------------+------------

Sample table: soccer\_country

country\_id | country\_abbr | country\_name

------------+--------------+---------------------

Sample table: match\_details

match\_no | play\_stage | team\_id | win\_lose | decided\_by | goal\_score | penalty\_score | ass\_ref | player\_gk

----------+------------+---------+----------+------------+------------+---------------+---------+-----------

**Solution:**

SELECT ass\_ref\_name, country\_name

FROM asst\_referee\_mast a

JOIN soccer\_country b

ON a.country\_id=b.country\_id

JOIN match\_details c

ON a.ass\_ref\_id=c.ass\_ref

WHERE play\_stage='F';

7) You have two SQL tables. The first one is called employees and it contains the employee names, the unique employee ids and the department names of a company.

Sample:

department\_name employee\_id employee\_name

The second one is named salaries. It holds the same employee names and the same employee ids ñ and the salaries for each employee.

Sample:

salary employee\_id employee\_name

The company has 546 employees, so both tables have 546 rows.

Print every department where the average salary per employee is lower than $500.

**Solution:**

SELECT department\_name, AVG(salaries.salary) AS avg\_salaries

FROM employees

JOIN salaries

ON employees.employee\_id = salaries.employee\_id

GROUP BY department\_name

8) You have two SQL tables: authors and books.

authors:

author\_name book\_name

books:

book\_name sold\_copies

Create an SQL query that shows the TOP 3 authors who sold the most books in total.

**Solution:**

SELECT authors.author\_name, SUM(books.sold\_copies) AS sold\_sum

FROM authors

JOIN books

ON books.book\_name = authors.book\_name

GROUP BY authors.author\_name

ORDER BY sold\_sum DESC

LIMIT 3;

9) Write a SQL statement to make a list in ascending order for the customer who works either through a salesman or by own.

Sample table: customer

customer\_id cust\_name city grade salesman\_id

----------- ------------ ---------- ---------- -----------

Sample table: salesman

salesman\_id name city commission

----------- ---------- ---------- ----------

Questions 10 through 14 use the following tables employees and department.

Sample table: employees

emp\_id | emp\_name | job\_name | manager\_id | hire\_date | salary | commission | dep\_id

Sample table: department

dep\_id | dep\_name | dep\_location

**Solution:**

SELECT a.cust\_name,a.city,a.grade,

b.name AS "Salesman",b.city

FROM customer a

LEFT JOIN salesman b

ON a.salesman\_id=b.salesman\_id

order by a.customer\_id;

10. Group average salary by job\_name

**Solution:**

SELECT job\_name,

avg(salary)

FROM employees

GROUP BY job\_name;

11. Count the number of unique job names.

**Solution:**

SELECT COUNT(DISTINCT job\_name)

FROM employee;

12. Assume 10% tax rate. Create a column that is the salary after taxes.

**Solution:**

ALTER TABLE employee

ADD Total INT NOT NULL DEFAULT 0

INSERT INTO employee(Total)

SELECT 0.9\*salary FROM employee;

13. Count all of employee ids that contain a job name the word 'data' in it.

**Solution:**

SELECT COUNT(emp\_id)

FROM employees

WHERE job\_name LIKE '%data%';

14. Subselect columns using a subquery.

**Solution:**

Subquery to Calculate Average salary –

SELECT emp\_id,

emp\_name,

(SELECT AVG(salary)

FROM employee) AS AverageSalary

FROM employee;

14. Why create views?

**Solution:**

1) Views are used for security purposes because they provide encapsulation of the name of the table. Data is in the virtual table, not stored permanently. Views display only selected data.

The syntax for creating a View is given below:

Create View Viewname As

Select Column1, Column2 From Tablename

Where (Condition) Group by (Grouping Condition) having (having Condition)

2) Views can hide complexity

If you have a query that requires joining several tables, or has complex logic or calculations, you can code all that logic into a view, then select from the view just like you would a table.

3) Views can be used as a security mechanism

A view can select certain columns and/or rows from a table, and permissions set on the view instead of the underlying tables. This allows surfacing only the data that a user needs to see.

4) Views can simplify supporting legacy code

If you need to refactor a table that would break a lot of code, you can replace the table with a view of the same name. The view provides the exact same schema as the original table, while the actual schema has changed. This keeps the legacy code that references the table from breaking, allowing you to change the legacy code at your leisure.

15. Explain to an eight year old (i.e. your professor) what are the first three Normal Forms.

**Solution:**

First normal form (1NF)

• Each table has a primary key: minimal set of attributes which can uniquely identify a

record

• The values in each column of a table are atomic (No multi-value attributes allowed).

2

• There are no repeating groups: two columns do not store similar information in the

same table.

Second normal form (2NF)

• All requirements for 1st NF must be met.

• No partial dependencies.

• No calculated data

Third normal form (3NF)

• All requirements for 2nd NF must be met.

• Eliminate fields that do not directly depend on the primary key; that is no transitive

dependencies.

Explain above using example.

16. What is an index? Why do we use them?

**Solution:**

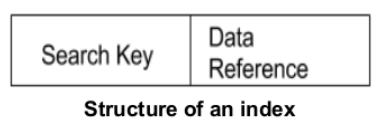
An index or database index is a data structure which is used to quickly locate and access the data in a database table. It is used to speed up queries.

Indexes are created using some database columns.

The first column is the Search key that contains a copy of the primary key or candidate key of the table. These values are stored in sorted order so that the corresponding data can be accessed quickly (Note that the data may or may not be stored in sorted order)

.

The second column is the Data Reference which contains a set of pointers holding the address of the disk block where that particular key value can be found



Uses of Indexes -

1. Support for fast lookup - Indexing is a way to optimize performance of a database by minimizing the number of disk accesses required when a query is processed.
2. Policing the database constraints - Indexes are used to police database constraints, such as UNIQUE, EXCLUSION, PRIMARY KEY and FOREIGN KEY. An index may be declared as UNIQUE, which creates an implicit constraint on the underlying table. Database systems usually implicitly create an index on a set of columns declared PRIMARY KEY.

Many database systems require that both referencing and referenced sets of columns in a FOREIGN KEY constraint are indexed, thus improving performance of inserts, updates and deletes to the tables participating in the constraint.